

comprising:

a computer readable medium; and

instructions stored on the computer readable medium to:

translate a plurality of source binary code instructions utilizing compiler annotation, the plurality of source binary code instructions are an executable and linking format (ELF) binary code file and the compiler annotation is an ELF section; and  
output a plurality of target binary code instructions.

37. (Original) The apparatus as recited in Claim 36, wherein the compiler annotation enables the translating the plurality of source binary code instructions to be performed on the plurality of source binary code instructions using a non-heuristic approach.

38. (Original) The apparatus as recited in Claim 36, wherein the compiler annotation describes functional characteristics of the plurality of binary code instructions.

39. (Original) The apparatus as recited in Claim 36, wherein the compiler annotation comprises one or more records selected from a module identification (ID), a function ID, a split function ID, a jump table ID, a function pointer initialization ID, a function address assignment ID, an offset expression ID, a data in the text section ID, a volatile load ID, and an untouchable region ID.

40. (Original) The apparatus as recited in Claim 36, wherein the instructions to translate the plurality of source binary code instructions comprises instructions to:

utilize the compiler annotation to partition the plurality of source binary code instructions into sections, functions and basic blocks; and  
build a control-flow graph utilizing the plurality of source binary code instructions and the compiler annotation.

#### REMARKS

Claims 1 - 5, 7 - 12, and 14 - 40 are pending in the application. Claims 1 - 5, 7 - 12, and 14 - 40 have been rejected. No new claims have been added.

Claims 1, 2, 3, 5, 8, 9, 10, 12, 16, 17, 18, 20-23, 25 - 28 and 30 - 40 stand rejected under Chen, U.S. Patent No. 6,625,807 (Chen) in view of Zucker, U.S. Patent No. 5,991,871 (Zucker) and further in view of Haungs, Executable and Linking Format (ELF), published on the Internet ([www.cs.ucdavis.edu/~haungs/paper/node10.html](http://www.cs.ucdavis.edu/~haungs/paper/node10.html)) (Haungs) . This rejection is respectfully traversed.

The present invention generally relates to an optimizing compiler that adds annotation information (i.e., compiler annotation) to an executable binary code file. The compiler annotation provides information useful for binary translators such that a binary translator does not have to use a heuristic approach to translate binary code. Compiler annotation identifies such information as function boundaries, split functions, jump table information, function addresses and code labels. (See e.g., application page 3, liens 1 – 9.)

More specifically, the present invention, as set forth by independent claim 1, relates to a method of producing a binary code file which includes compiling a plurality of source code instructions, and outputting a plurality of binary code instructions and compiler annotation. The plurality of binary code instructions is executable by a processor of a computer system and are an executable and linking format (ELF) binary code file. The compiler annotation is an ELF selection.

The present invention, as set forth by independent claim 8, relates to a method of translating a source binary code file which includes translating a plurality of source binary code instructions utilizing compiler annotation and the outputting a plurality of target binary code instructions. The plurality of source binary code instructions are an executable and linking format (ELF) binary code file and the compiler annotation is an ELF section. The plurality of target binary code instructions is executable by a processor of a computer system.

The present invention, as set forth by independent claim 16, relates to a binary code file which includes a plurality of binary code instructions and compiler annotation. The compiler annotation enables a binary translator to utilize the compiler annotation to partition the plurality

of binary code instructions into sections, functions and basic blocks; and build a control-flow graph utilizing the plurality of binary code instructions and the compiler annotation. The plurality of binary code instructions is executable by a processor of a computer system and the instructions are an executable and linking format (ELF) binary code file. The compiler annotation is an ELF section.

The present invention, as set forth by independent claim 21, relates to an apparatus for producing a binary code file which includes means for compiling a plurality of source code instructions and means for outputting a plurality of binary code instructions and compiler annotation. The plurality of source binary code instructions are an executable and linking format (ELF) binary code file and the compiler annotation is an ELF section.

The present invention, as set forth by independent claim 26, relates to an apparatus for translating a source binary code file which includes means for translating a plurality of source binary code instructions utilizing compiler annotation, and means for outputting a plurality of target binary code instructions. The plurality of source binary code instructions are an executable and linking format (ELF) binary code file and the compiler annotation is an ELF section.

The present invention, as set forth by independent claim 31 relates to an apparatus for producing a binary code file which includes a computer readable medium, and instructions stored on the computer readable medium to compile a plurality of source code instructions, and output a plurality of binary code instructions and compiler annotation. The plurality of source binary code instructions are an executable and linking format (ELF) binary code file and the compiler annotation is an ELF section.

The present invention, as set forth by independent claim 36, relates to an apparatus for translating a source binary code file which includes a computer readable medium, and instructions stored on the computer readable medium to translate a plurality of source binary code instructions utilizing compiler annotation, and output a plurality of target binary code instructions. The plurality of source binary code instructions are an executable and linking format (ELF) binary code file and the compiler annotation is an ELF section.

Chen discloses a method for register optimization during code translation and utilizes a technique that removes the time overhead for analyzing register usage and eliminates fixed restraints on the compiler register usage. The method for register optimization utilizes a compiler to produce a bit vector for each program unit (i.e., subroutine, function, and/or procedure). Each bit in the bit vector represents a particular caller-saved register. A bit is set if the compiler uses the corresponding register within that program unit. During the translation, the translator examines the bit vector to very quickly determine which registers are free, and therefore can be used during register optimization without having to save and restore the register values.

Zucker discloses interfacing a binary application program to a computer system. The application binary interface includes linkage structures for interfacing the binary application program to a digital computer. A function in a relocatable shared object module obtains the absolute address of a Global Offset Table (GOT) in the module using relative branch and link instructions through the computer's link register. A dynamic linker constructs a Procedure Linkage Table (PLT) and a pointer table for an object module in a process memory image in which space is allocated for the PLT, but the PLT is not initially provided. The pointer table stores absolute addresses of external functions that cannot be reached by relative branching from the module. The PLT receives calls to these functions, gets the absolute addresses from the pointer table and branches to the absolute addresses of the functions. The PLT also receives calls to functions that can be reached by relative branching from the module, and causes relative branching to the functions.

Zucker sets forth that a binary application object file program is configured in an Executable and Linking Format (ELF). More specifically, Zucker sets forth:

The object program 60 comprises an ELF header 64 that specifies the number and sizes of the sections of the program 60, in addition to other information required by the operating system 48. A section header table 66 is also provided, including information required to locate all of the sections of the file 60.

The actual text (program instructions) and data of the program are provided in a number of discrete sections, designated as 68<sub>1</sub> to 68<sub>N</sub>. A dynamic section 70 contains dynamic linking information for use by the dynamic linker 54 (Zucker Col. 8, lines 28 -39).

Huang sets forth a description of the executable and linking format. When discussing the ELF sections, Huang sets forth:

There are a number of types of sections described by entries in the section header table. Sections can hold executable code, data, dynamic linking information, *debugging data*, symbol tables, relocation information, comments, string tables, and notes. Some sections are loaded into the process image and some provide information needed in the building of a process image while others are used only in linking object files. Figures 2.9 displays a list of special sections along with a brief description (Huang, Emphasis added).

In general, Chen, Zucker and Huang do not disclose or suggest method of producing a binary code file where the binary code file includes binary code instruction and *compiler annotation* where the *compiler annotation is an ELF section*, as substantially required by each of the independent claims. The ELF header 64 of Zucker neither teaches nor suggests the claimed binary code file having both binary code instructions and compiler annotation which is an ELF section. Additionally, “debugging data” as set forth by Huang neither teaches nor suggests the claimed compiler annotation.

More specifically, Chen and Zucker, taken alone or in combination, do not teach or suggest a method of producing a binary code file which includes compiling a plurality of source code instructions, and outputting a plurality of binary code instructions and *compiler annotation* wherein the *plurality of binary code instructions is executable by a processor of a computer system and are an executable and linking format (ELF) binary code file and the compiler annotation is an ELF section*, all as required by claim 1. Accordingly, claim 1 is allowable over Chen and Zucker. Claims 2 - 7 depend from claim 1 and are allowable for at least this reason.

Chen and Zucker, taken alone or in combination, do not teach or suggest a method of translating a source binary code file which includes translating a plurality of source binary code instructions utilizing *compiler annotation* and the outputting a plurality of target binary code instructions wherein the plurality of source binary code instructions are *an executable and linking format (ELF) binary code file and the compiler annotation is an ELF section* and the plurality of target binary code instructions is executable by a processor of a computer system, all as required by claim 8. Accordingly, claim 8 is allowable over Chen and Zucker. Claims 9 - 15 depend from claim 8 and are allowable for at least this reason.

Chen and Zucker, taken alone or in combination, do not teach or suggest a binary code file which includes a plurality of binary code instructions and *compiler annotation* where the plurality of binary code instructions are an executable and linking format (ELF) binary code file and the *compiler annotation is an ELF section*, all as required by claim 16. Accordingly, claim 16 is allowable over Chen and Zucker. Claims 17 - 20 depend from claim 16 and are allowable for at least this reason.

Chen and Zucker, taken alone or in combination, do not teach or suggest an apparatus for producing a binary code file which includes means for outputting a plurality of binary code instructions and *compiler annotation* wherein the plurality of source binary code instructions are an executable and linking format (ELF) binary code file and *the compiler annotation is an ELF section*, all as required by claim 21. Accordingly, claim 21 is allowable over Chen and Zucker. Claims 22 - 25 depend from claim 21 and are allowable for at least this reason.

Chen and Zucker, taken alone or in combination, do not teach or suggest an apparatus for translating a source binary code file which includes means for translating a plurality of source binary code instructions utilizing *compiler annotation*, and means for outputting a plurality of target binary code instructions wherein the plurality of source binary code instructions are an executable and linking format (ELF) binary code file and *the compiler annotation is an ELF section*, all as required by claim 26. Accordingly, claim 26 is allowable over Chen and Zucker. Claims 27 - 30 depend from claim 26 and are allowable for at least this reason.

Chen and Zucker, taken alone or in combination, do not teach or suggest an apparatus for producing a binary code file which includes instructions stored on a computer readable medium to compile a plurality of source code instructions, and output a plurality of binary code instructions and *compiler annotation* wherein the plurality of source binary code instructions are an executable and linking format (ELF) binary code file and *the compiler annotation is an ELF section*, all as required by claim 31. Accordingly, claim 31 is allowable over Chen and Zucker. Claims 32 - 35 depend from claim 31 and are allowable for at least this reason.

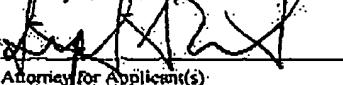
Chen and Zucker, taken alone or in combination, do not teach or suggest an apparatus for translating a source binary code file which includes instructions stored on a computer readable medium to translate a plurality of source binary code instructions utilizing *compiler annotation*,

and output a plurality of target binary code instructions wherein the plurality of source binary code instructions are an executable and linking format (ELF) binary code file and the *compiler annotation is an ELF section*, all as required by claim 36. Accordingly, claim 36 is allowable over Chen and Zucker. Claims 37 - 40 depend from claim 36 and are allowable for at least this reason.

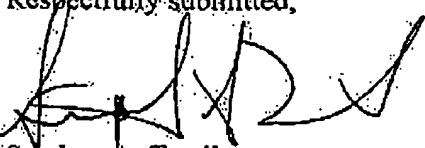
Additionally, Chen, Zucker and Huangs, taken alone or in combination do not disclose or suggest that the compiler annotation *enables binary translation to be performed on the plurality of binary code instructions using a non-heuristic approach*, as required by claims 2, 9, 17, 22, 27, 32, and 37. Additionally, Chen, Zucker and Huangs, taken alone or in combination do not disclose or suggest that the compiler annotation *describes functional characteristics of the plurality of binary code instructions*, as required by claims 3, 10, 18, 23, 28, 33, and 38.

### CONCLUSION

In view of the amendments and remarks set forth herein, the application is believed to be in condition for allowance and a notice to that effect is solicited. Nonetheless, should any issues remain that might be subject to resolution through a telephonic interview, the examiner is requested to telephone the undersigned.

I hereby certify that this correspondence is being sent to the Commissioner for Patents via the USPTO Central Facsimile on February 2, 2006.	
	<u>2/2/06</u>
Attorney for Applicant(s)	Date of Signature

Respectfully submitted,

  
Stephen A. Terrile  
Attorney for Applicant(s)  
Reg. No. 32,946